Generating Product-Quality SDKs and API Reference Documentation

Mike Kistler, Allen Dean, Logan Patiño, Dustin Popp, German Attanasio Ruiz, Taj Santiago, Jenifer Schlotfeldt, Jeffrey Stylos
IBM
Motivation

**API Doc**

```
"VersionQueryParam": {
  "in": "query",
  "name": "version",
  "required": true,
  "default": "2018-07-10",
  "pattern": "^\d{4}-\d{2}-\d{2}$",
  "type": "string",
  "description": "Release date of the API version you want to use."
},
```

**Java SDK**

```java
Assistant service = new Assistant("2018-07-10");
WorkspaceCollection workspaces = service.listWorkspaces().execute();
```
Goals

- **Time to Market:** Decrease time from new features available in a service and the SDK support for these features

- **Quality:** Improve quality by imprinting best practices into the code and avoiding common errors that occur in hand-written SDKs and API reference

- **Compliance:** Improve/enforce agreement between the API definition (Swagger) and SDKs / API reference

- **Consistency across services:** Common style for a particular language SDK across a family of APIs

- **Consistency across languages:** Improve consistency of structure/style/naming across SDKs to support knowledge transfer across languages
Options parameters (Java, Go, Apex)

Java Client Library

```java
public ToneAnalysis tone(
    ToneInput toneInput,
    Boolean sentences,
    List<String> tones,
    String contentLanguage,
    String acceptLanguage)
```

Application Code

```java
ToneAnalysis response = tone(
    myToneInput,
    null,
    null,
    null,
    null);
```
Options parameters (Java, Go, Apex)

**Java SDK**

```java
public ToneAnalysis tone(ToneOptions toneOptions)
```

**Application Code**

```java
ToneOptions toneOptions = ToneOptions.Builder()
    .toneInput(myToneInput)
    .build();
ToneAnalysis response = tone(toneOptions);
```
File Parameters

API Doc

```
"filePart": {
    "name": "file",
    "in": "formData",
    "type": "file",
    "description": "The content of the document to ingest."
},
```

Java SDK

```java
public class AddDocumentOptions extends GenericModel {
    private InputStream file;

    public static class Builder {
        public Builder file(InputStream file) { ... }
        public Builder file(File file) { ... }
    }
}
```
File Parameters

API Doc

```
"filePart": {
    "name": "file",
    "in": "formData",
    "type": "file",
    "description": "The content of the document to ingest.",
    "x-include-filename": true,
    "x-file-content-types": [
        "application/json",
        "application/msword",
        "application/pdf",
        "text/html"
    ]
},
```

Application Code

```java
AddDocumentOptions options = AddDocumentOptions.Builder()
    .file(documentStream).filename(documentName)
    .fileContentType("application/pdf").build();
DocumentStatus status = discovery.addDocument(options);
```
More special cases for SDK generation

- Collapse references to primitive type models and array models into their reference sites
- Special handling for composed models to compose at the level of property attributes
- Discard explicit `content-type` or `accept-type` parameter and instead generate these when appropriate
- "Explode" a body model parameter into a individual parameters for each body model property
- Special processing for operations that support body content of multiple types
- For file parameters, generate a separate, possibly required, filename parameter
- For file parameters that support multiple content types (specified using an annotation), generate an associated content-type parameter.
- For strongly typed languages, generate separate methods for each "produces" type of an operation.
- Allow request bodies and file type parameters to be passed as either streams, files, or as simple variables.
More special cases for SDK generation

- Exclude some operations from the generated SDK
- Special handling for models with `additionalProperties`
- Preprocessing for API Reference generation
- Special handling of markdown fields to preserve embedded newlines (which are significant in markdown)
- Generate service methods in the same order as their appearance in the API definition
- Prune unneeded models from the SDK
- Careful formatting and flow of comments to present embedded documentation in a readable fashion.
- Generate constants for string enum values
- Support a means for supplying headers to be sent with each request and also on a per-request basis.
- All SDKs are designed to allow the user to access headers returned in the response
Quality of Generated SDKs and Docs

- IBM API Guidelines
  - Response schema should not be bare array
  - Property and parameter names should be lower-snake-case
- OpenAPI Style Guide - conventions/best practices for SDK generation
  - Every operation should have a (unique) operationId
  - Schema element should use well-defined type/format
  - Required parameters should appear before optional parameters
  - All operations/parameters/models/properties should have a description
  - Schema in response should reference a named model
- OpenAPI Validator/Linter
  - Configurable & extensible validator/linter for OpenAPI docs
- Compatibility rules for SDK generation
SDK generation tooling

**watson-swagger-codegen**

- Languages (9)
- Templates
- WatsonCodegen

**swagger-api**

- swagger-codegen
- swagger-core
- swagger-parser

Boxes roughly proportional to code size
SDK generation tooling

- Parse API doc into tree of API doc objects
  - Preprocess API doc object tree
  - Create "codegen" objects for API elements
  - Postprocess codegen objects
  - Generate code w/ template engine

- Processing organized as multiple transformation passes
  - Create options param
  - Handle accept and content-type params
  - Add associated params for files
  - Collapse refs to array and primitive type models

- Language-specific types and names assigned
  - Assign alternate names
  - Process Enum values

- Transform codegen objects
  - Order service methods by tag and position in API doc
  - Prune unneeded models
  - Collect information for API reference
Accomplishments — Watson Developer Cloud

Generated SDKs for 10 services and 9 languages
- Java, Node, Python, Swift, DotNet (C#), Ruby, Go, Apex, IBM Cloud fns
- 98% generated -- exceptions are web sockets and some special methods
- Currently on roughly two week release cycle

The OpenAPI Validator/Linter
- Used as PR merge quality gate
- Has identified many API definitions that have since been fixed

Generated API reference
- API reference that puts SDKs in the fore-front
- Phase I implementation in production for Watson services
- Phase II implementation for all IBM Cloud: > 66 services currently
Still work to be done

- Support for OpenAPI 3.0 API documents
- Generate test cases -- this is currently a completely manual process
- Websockets
References

Watson SDKs on GitHub
  • https://github.com/watson-developer-cloud

Watson SDK/API Reference (generated)
  • https://console.bluemix.net/developer/watson/documentation

Watson REST API Guidelines
  • http://watson-developer-cloud.github.io/api-guidelines/

Swagger Coding Guidelines
  • http://watson-developer-cloud.github.io/api-guidelines/swagger-coding-style
Questions?